A user supported object tracking framework for interactive video production

Christian Bailer, Alain Pagani, Didier Stricker

Augmented Vision
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Str. 122, Kaiserslautern, Germany

email: {Christian.Bailer | Alain.Pagani | Didier.Stricker}@dfki.de

Abstract

We present a user supported tracking framework that combines automatic tracking with extended user input to create error free tracking results that are suitable for interactive video production. The goal of our approach is to keep the necessary user input as small as possible. In our framework, the user can select between different tracking algorithms – existing ones and new ones that are described in this paper. Furthermore, the user can automatically fuse the results of different tracking algorithms with our robust fusion approach. The tracked object can be marked in more than one frame, which can significantly improve the tracking result. After tracking, the user can validate the results in an easy way, thanks to the support of a powerful interpolation technique. The tracking results are iteratively improved until the complete track has been found. After the iterative editing process the tracking result of each object is stored in an interactive video file that can be loaded by our player for interactive videos.

First presented at the 11th European Conference on Visual Media Production CVMP 2013, extended and revised for JVRB

Keywords: interactive tracking, interactive video, web applications, data fusion, tracking methods

1 Introduction

Interactive videos in which objects can be clicked by the user have many advantages over ordinary videos, as they allow users to get additional information about video content in a simple and intuitive way. This makes them interesting for many different applications in the field of advertisement, entertainment and education. For example, educational videos can contain additional sources of knowledge inside the interactive content, or promotional videos can include invisible advertisement that appears on demand, i.e. when the user is interested.

At the same time, the development of new devices like smartphones, tablets or Smart TVs generalizes the concept of video watching to a more interactive relation where the user gets more involved. However, the production of such videos is challenging and time consuming, as each object has to be marked in each frame where it is visible. While an obvious solution would be to employ an automatic tracking algorithm to follow the objects in the sequence, this proves to be unreliable as even state-of-the-art tracking methods cannot deal with many situations that occur in practice (like strong appearance changes). Comparison papers like Wang et al. [WCXY11] have shown that modern tracking methods are still error-prone and that their reliability is very sequence dependent – it is even possible that a tracking algorithm that performs good on many difficult sequences fails on an easy sequence. As a result, even if we use automatic object tracking methods there can be a lot of manual postprocessing work necessary to fix tracking errors.

In this work we look into this problem and aim to create a semi-automatic tracking approach that minimizes the human effort necessary for creating interactive videos. From the perspective of the tracking problem this means that we want to create error free tracking results, suitable for interactive videos, with as little user input as possible. This contrasts with the standard tracking approach where the goal is to automatically produce the best possible tracking result for a minimal amount of user input (usually one object bounding box in the very first frame).

Figure 1 shows a possible guideline for our semi-automatic tracking pipeline. Our paper contains several important contributions: First, we show that reliable tracking results cannot be achieved by automatic trackers and some kind of user interaction is required when a perfect result is mandatory. We therefore investigate ways to guarantee perfect tracking results while minimizing the human effort for interactive video production. For that reason, we first analyze the possibilities offered by having user input over multiple frames. Furthermore, we present a method for fusing the results of independent trackers and show in our evaluation that the fusion outperforms the best trackers. In addition, we present a new particle tracking method and three new specialized trackers for specific scenarios. Also, we suggest a new way of evaluating the robustness of trackers by comparing their outputs depending on the number of input frames. For the user interface, we present a new validation tool based on motion interpolation that drastically reduces the time required to validate a tracking result. Altogether, we demonstrate a complete web system (named OnEye) for interactive video production, design and playback.

The remainder of this paper is as follows: Section 2 reviews related work in terms of automatic and user-supported tracking methods. The tracking itself is discussed in Section 3. Section 4 describes our user interface for user supported creation of interactive videos and Section 5 gives a description of our full OnEye system. In Section 6 we present the results of our evaluation. Concluding remarks are provided in Section 7.

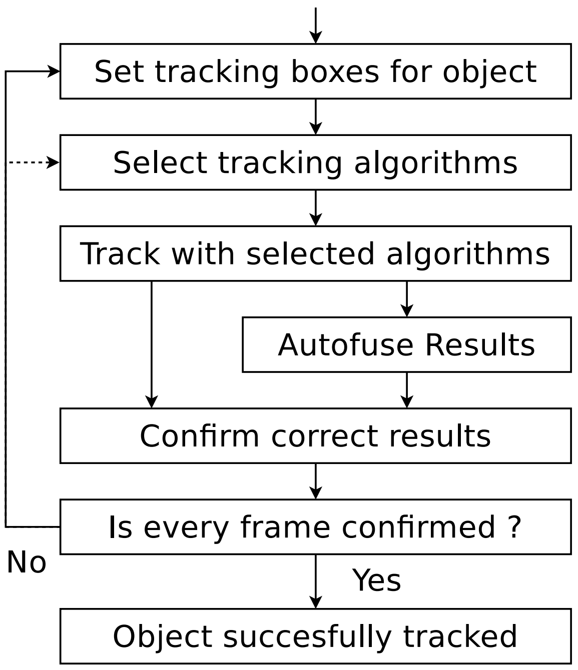

Set tracking boxes for object
Select tracking algorithms
Track with selected algorithms
Autofuse Results
Confirm correct results
Is every frame confirmed ?
No
Yes
Object succesfully tracked

Figure 1: A guideline for our semi-automatic tracking pipeline. The user can iteratively improve the tracking result, by setting additional tracking boxes at frames where tracking failed or by tracking with additional algorithms.

2 Related Work

Due to extensive research much progress has been made in automatic object tracking in the last years. An overview over automatic methods can be found in surveys like [YSZ+11, CAG12] or tracking evaluation publications like [WCXY11, WLY13]. In the most recent evaluation of Wu et al. [WLY13] the on average best out of 29 tested algorithms SCM [ZLY12] gained an average overlap to the ground truth of around 50 % on the tested sequences. In around 25 % of the frames it achieved an overlap of at least 80 % (which we consider as sufficient for our application).

Despite these good results a naive approach would still require a lot of postprocessing work to gain a sufficient quality for all frames. Nevertheless, in literature little to no effort is put into the question how modern tracking algorithms can be utilized to create an error free tracking result with as little user effort as possible. However, there are publications dealing with the question in general – without directly utilizing modern tracking algorithms.

Vondrick et al. [VPR13] created a framework for fast video annotation. The user has to select the object in several but not all frames as their approach is able to interpolate the trajectory in frames between two user labeled frames. For interpolation they also consider image appearance to find a trajectory that is short in image space but still has an appearance similar to the user selected object. We think that this idea is interesting to get a more accurate trajectory if only a few frames have to be bridged. However, for longer trajectories the “short path” constraint in image space will probably be a drawback as it limits the freedom of movement of the curve too much. With modern tracking algorithms we expect to be often able to track a lot of frames at once. In their work Vondrick et al. also provide user studies and statics concerning workers that are practically using their software. In their previous work [VR11] they present an extension of their interpolation approach based on active learning. The extension is not meant to improve the interpolation result itself. Instead it utilizes the interpolation energy function to find the frame between two user set frames that is most uncertain in its position according to their interpolation model. The idea is that the user only labels very few frames and the application then tells the user which frame to label next. While the idea sounds interesting they did not integrate it into their later work [VPR13] as there are still some unresolved user interface issues.

There are also publications for semi-automatic video annotation that are based on the exact object boundary or pixel-level labeling instead of rectangular boxes (which are used in the vast majority of automatic tracking methods). The approach of Bertolino et al. [Ber12] is based on the segmentation of objects. The user’s task is to initialize the segmentation and to correct it if it gets erroneous over time. To fulfill the task the application provides the user several frame based editing tools. Yuen et al. [YRLT09] designed a video annotation system that allows people to annotate objects in a web browser. They also use techniques like interpolation between user labeled frames. The approach of Vijayanarasimhan et al. [VG12] tries similar to [VR11] to find the frames the user has to label to minimize the error for non user labeled frames. However, it is designed for pixel-level labeling. A general drawback of all boundary or pixel-level labeling based approaches is that the higher accuracy will always result in a clearly larger human effort compared to bounding box based approaches. We think that the higher accuracy is not worth the additional effort for most interactive video applications.

3 Tracking with extended input

In this section we describe our automatic tracking approach that can create clearly better tracking results than common approaches, whenever there is extended user input available. Usually, tracking algorithms only use one object bounding box at the first frame as input. This gives the tracker only the minimal information necessary for tracking. In contrast, our approach can use additional bounding boxes at arbitrary frames. We call the bounding boxes provided by the user as reliable input *user set bounding boxes*. Setting a few more bounding boxes is of no big effort for the user but can improve tracking results significantly.

Furthermore, the tracking algorithm is not fixed in our approach, but the user can rather select a specific tracking algorithm depending on his/her experience and the sequence at hand. Moreover and more importantly, the user can select more than one algorithm for tracking. The results of the tracking algorithms can then be fused automatically (Section 3.4) or manually (Section 4).

3.1 Tracking methods

As no automatic tracking methods performs reliably for all kind of sequences [WCXY11], we implemented several different methods with different strengths and weaknesses. We divide them into two groups: General methods and specialized methods that beat the general methods in special situations.

3.1.1 General methods

The general tracking methods we implemented are P-Channel [PSF09], a modified version of the MILTrack algorithm [BYB11] with HAAR and HOG features, Visual Tracking Decomposition (VTD) [KL10] and Circulant Structure with Kernels (CSK) [HCMB12].

We use HAAR and HOG features together in the MILTrack classifier as we found this to work in average better than one feature type only. Additionally, we use for MILTrack and P-Channel a particle based motion model. This gives a much better robustness than a simple motion model. Each particle represents a possible bounding box for the object. In the initial frame there is only one particle for the user set bounding box. For the second frame N new particles are spread from the initial particle, by Gaussian distributions for 2 up to 4 dimensions of the bounding box: x and y position and if desired also x and y scale (see Section 3.3). In order to avoid a tedious parameter tuning, the spread variances are not absolute but relative to the object size. Furthermore, we deal with unusual high accelerations by using two different spread variances in the motion model, with 50 % of the particles, each. The bigger variance can handle high accelerations, while the smaller one avoids drifting because of high particle density with low acceleration. This approach is similar to the dual motion model of [KL10]. The bounding boxes of the spread particles are then tested for the likeliness that they represent the object by the MILTrack or P-Channel appearance model. We call this value *raw probability*. Usually the raw probability does not represent the real probability and is not strict enough in penalizing bad positions. As a result particles would spread to far when using the raw probability directly for particle spreading. To avoid this we transfer the raw probability L_r into a stricter *particle probability* L_p:

$$L_u(p) = e^{(\gamma L_r(p))} - 1 \tag{1}$$

$$L_p(p_i) = L_u(p_i) / \sum_{p_j \in P} L_u(p_j) \tag{2}$$

P is the set of all particles. L_u is a function that puts the probabilities into the exponential space. With an appropriate γ this deprecates particles drastically compared to particles with better raw probability. γ controls the extend of the depreciation. A bigger γ leads to more depreciation. In practice, γ controls mainly the particle spread. We borrowed this idea from [KC11]. Different to [KC11] we subtract "1" to get $L_u = 0$ for $L_r = 0$. This gives us slightly better results. Without this, small probabilities are overrated. In L_p we are normalizing L_u so that all all particles together have the probability 1. While in the first frame all particles spread from the initial particle, particles in following frames can spread from any existing particle. The probability that a particle p_{new} spreads from a particle p_{old} is $L_p(p_{old})$. Thus, with a big γ the particles are kept close to the best appearance, with a medium γ it is possible to pursue different hypothesis and with a small γ particles will spread far. 20 % of the particles are directly spread from the position of the parent particle, while for 80 % of the particles the velocity created by the parent particle when spreading from the grand parent particle is added as well. As a consequence, 80 % of the particles are acceleration hypotheses and the current speed is given by the parent. Acceleration hypotheses are usually advantageous over speed hypotheses. However, there are cases like a sudden object stop where speed hypotheses perform better. Therefore, 20 % of the particles follow a speed hypotheses. Note that particles are not inheriting their spread variances or hypotheses model from their parents.

To determine the object bounding box for a frame we do not just take the bounding box of the best particle. Instead we add all particles with a center location close to that of the best particle, namely within 20 % of the best particle's bounding box size, to a cluster C_1. Then we create the clusters C_2 to C_5 in the same manner. The best particle for these clusters is here the best particle that is not yet in a cluster. We then take the cluster C_{best} with the highest sum of particle probabilities L_p as the final cluster used for object bounding box creation.

$$C_{best} = \arg\max_{n=1...5} \sum_{p \in C_n} L_p(p) \tag{3}$$

The final object bounding box B_f is then:

$$B_f = \sum_{p_1 \in C_{best}} B(p_1) L_p(p_1) / \sum_{p_2 \in C_{best}} L_p(p_2) \tag{4}$$

where $B(p)$ is the bounding box of a particle (Namely, its x, y,width and height parameters written as vector). The averaging leads to a more robust object position, as it not only considers the locally best position according to the classifier but also the classifier results for positions close to that position. If there are for example more good particles to the left of the best particle than to the right, the exact object position is likely left to the best particle. Note that particles with moderate or low raw probability have virtually no influence in averaging with appropriate γ as their particle probability is close to zero.

3.1.2 Specialized methods

We implemented three specialized methods. The first one is a color based tracker that is extremely reliable if background and foreground consist of different colors. The second one is a template based tracker which clearly outperforms common state of the art trackers in sequences like in Figure 2. The third one is a blob tracker that works only with static background. The blob tracker can track many small and fast objects (see Figure 3), where other generic trackers fail completely.

Color based tracker For the color based tracker we create two color histograms. One for the pixels inside the object's bounding box (foreground) and one for pixels around the object's bounding box (background). To deal with illumination changes each pixel votes not only for its color bin in the histogram but also with a lower weight for neighboring bins. We choose a bin size of 4x4x4 in RGB space which gives 64x64x64 bins. Pixels vote with a weight of 4 for their own bin and with a weight $max(0, 4-d)$ for neighboring bins of distance d. To get color ratings we subtract the background histogram from the foreground histogram. Thereby colors that are mainly in the background get a negative and colors that are mainly in the foreground a positive value in the subtracted histogram that serves as look-up table to rate colors. To find the object's bounding box we search the region that maximizes the sum of pixel ratings for the pixels inside the bounding box.

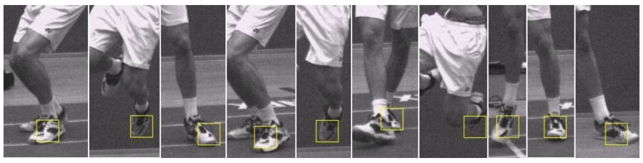


Figure 2: Intelligent template tracking with proper heuristics can be very effective for suitable sequences. The shoe is successfully tracked for a long time although there are many difficult situations and there is no object redetection i.e. pure tracking.

Template tracker Template tracking is old but still often used tracking approach. An important problem in template tracking is the template update problem [MIB04]. The template should be updated from time to time when the object appearances changes to avoid loosing the object. However, template updates cause drifts of the template compared to the object center that eventually will lead to object loss. We found that we can avoid drifting on suitable sequences by using several templates instead of only one. We create a new template every 25 frames starting from the first frame and keep up to 10 of them. If we have more, we remove the oldest one from our template list. To track the object we calculate the normalized cross correlation score for all (up to) 10 templates to all reasonable positions in the target image. Then we sum up all 10 scores for each position. The object's bounding box is then simply located at the position with the highest sum of scores. Reasonable positions in the target image are all positions that are lying within one bounding box size around the objects position in the previous frame shifted by the previous velocity. This approach can effectively avoid drifting as can be seen in Figure 2.

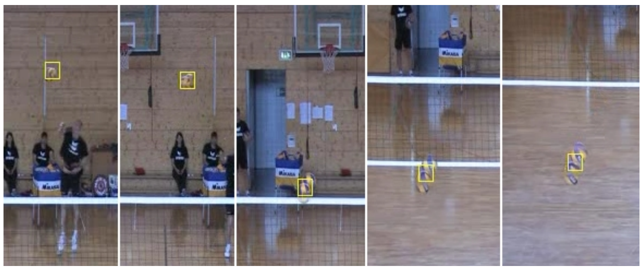


Figure 3: A simple blob tracker can beat advanced trackers with static background. Note: The background does not move. We just show different parts of the overall image.

Blob tracker For the blob tracker we first perform background subtraction between the current and last image with the approach described in [Ziv04]. Then we erode the result one time and dilate it 3 times with a 3x3 kernel. This yields more stable noise free blobs. After that we create a binary image out of the result by treating every value grater than zero as “1”. Connected regions of pixel value “1” we consider as blobs. If there is exactly one blob that has at least 50 % and at maximum 166 % of the blob size in the last frame the blob specifies the object. Otherwise we consider it as lost.

3.2 Benefiting from additional user input

There are several ways to benefit from more than one input frame in a tracking sequence. Simply resetting the tracking box when tracking through a user set frame already can avoid long term tracking failure as can be seen in Figure 4b. Furthermore, between two user set frames it is also possible to track the first half forward and the second half backward (Figure 4d), which further reduces the amount of tracking failures.

An even better strategy is to track the sequence completely forward and backward first, and fuse the results of the two tracking directions after tracking (Figure 4e). Because of the different temporal tracking directions it is very unlikely that the trackers of both directions follow the same wrong tracking path. The reason can be seen in Figure 5. If the forward tracker looses the object (1a,1b) on the path 1a$\rightarrow$2b it is of course possible to loose it with backward tracking as well, but the same wrong path 2b is impossible for the backward tracker. Only in the unlikely situation that the wrong path gets back to the correct one like in

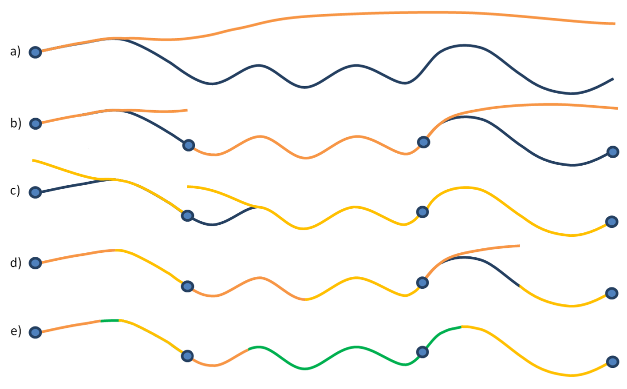


Figure 4: Different tracking strategies. a) One starting frame only b) Forward c) Backward d) Half forward and backward e) Forward and backward with fusion. Blue is the object movement, orange and yellow are forward and backward tracked segments and green are fused/“reliable” segments.

3b and the backward tracker also decides for that path both paths are identical.

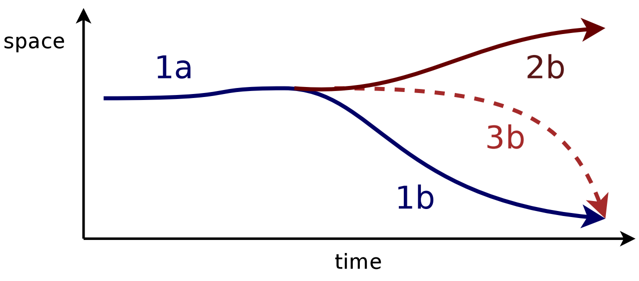


Figure 5: Tracking paths. Only the correct path 1b and the unlikely path 3b can be tracked forward as well as backward. 2b is impossible to be tracked backward as the backward tracker starts at the end of 1b.

Therefore, if the overlap (Equation 7) of the tracking boxes of both tracking directions is for a frame above a threshold α we assume that both tracking boxes roughly match the bounding box of the tracked object. The fused result at such a frame is set to the average tracking box of both directions and the frame is marked as “reliable”. If the overlap is below α we assume that at least the tracking result of one direction must be erroneous and we have to decide which one. We call such frames uncertain frames. Connected uncertain frames we call uncertain segments.

For uncertain segments that lie between a user set and a “reliable” frame we choose the tracking direction from the user set towards the “reliable” frame to be more reliable. The reason is that we know for this direction that the error is zero at the user set frame and likely small at the “reliable” frame, while it is likely small at the “reliable” frame but big (at least ¿α) at the user set frame for the other tracking direction. Obviously, the direction where the error is small at both ends should nearly always be the better one. It can also happen that a non “reliable” frame lies between two reliable frames. Here, it is more difficult to find a measure for the better tracking direction, as we have no user set frame at hand. We tried to use the internal rating value of the tracker, that is used to find the object, to decide which direction is better. However, this turned out to be not very reliable, which is no big surprise as the tracking failed because of the unreliability of this value. Instead we compare the results of the tracking directions to the interpolation created by the interpolation approach introduced in Section 4.1 and take the direction that is more similar to the interpolation (Different to Section 4.1 we also considers “reliable” frames for interpolation). Note that it is no good idea to directly use the interpolation approach in the motion model of a tracker as this will negatively influence the tracking result for frames where tracker and interpolation disagree. However, for selecting the better tracking direction it is advantageous as long as the interpolation is closer to the correct result than to a random tracking failure which should be fulfilled in most cases. If there are no ”reliable“ frames between two user set frames we trust both directions for 50 % of the way like in Figure 4d.

3.3 Dealing with scaling

Objects might not only move in the video but also change their size in pixels. However, a tracking approach that explicitly considers scale variations has more degrees of freedom and is therefore less stable. This is the reason why it is often ignored by standard tracking approaches. In our application we let the user choose which model of scale variation to use. The three basic models we propose are no scale variation, fixed ratio variation and free size change where the width and height of the object can change independently. Furthermore, we use an interpolated size model, which uniformly interpolates the size between two user set frames. It is the most important model in our application as it is in most situations sufficient to model the object size, while avoiding additional degrees of freedom.

3.4 Fusing results of different trackers

If a user tracks a sequence with different trackers he/she creates different hypotheses for the correct tracking result. He/She can then check manually these hypotheses and transfer the correct parts of the single hypotheses into the final result. While this approach is working well, it means also a lot of effort for the user if the good parts are strongly fragmented within different tracking results. Hence, we provide a more efficient alternative, where the different results are fused automatically into one result. The user checks in the first place only the fused result and only at frames where it is wrong he decides if it is worth to check also the source results or directly do something else like track again with more user set bounding boxes.

To fuse the different tracking results, we first find for each frame a reference tracking result r_f:

$$r_f = \underset{t \in T_f}{\operatorname{argmax}} \sum_{i \in T_f} O_\alpha(t,i)(\beta G(i) + 1) \tag{5}$$

where T_f are the $|T|$ tracking results for a frame f and $G(i)$ is 1 for frames that where marked as reliable in Section 3.2, 0 otherwise. $O_\alpha(t,i)$ is calculated as:

$$O_\alpha(t,i) = \begin{cases} 0 & O(t,i) < \alpha \\ O(t,i) & \text{otherwise} \end{cases} \tag{6}$$

$O(t,i)$ is the overlap between two tracking results calculated for two boxes B_1 and B_2 in general as:

$$O(B_1,B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2} \tag{7}$$

If r_f is not unique it is set to the track with the most "reliable" frames. This is useful if only one tracker is good for a sequence. When we have found r_f we average its bounding box with the bounding boxes of all tracking results that have an overlap of at least α i.e. where $O(r_f,i) \geq \alpha$. The weighing for the averaging is 1 for normal tracking boxes and 2 for reliable boxes because they already were averaged in Section 3.2. If at least 50 % of the boxes are averaged we mark the fused result as reliable. We count all reliable input boxes as $\beta + 1$ boxes and normal boxes as 1 box. The "reliable" marker of the fusion is not designed for further fusion but only as visual output for the user interface.

4 The User Interface

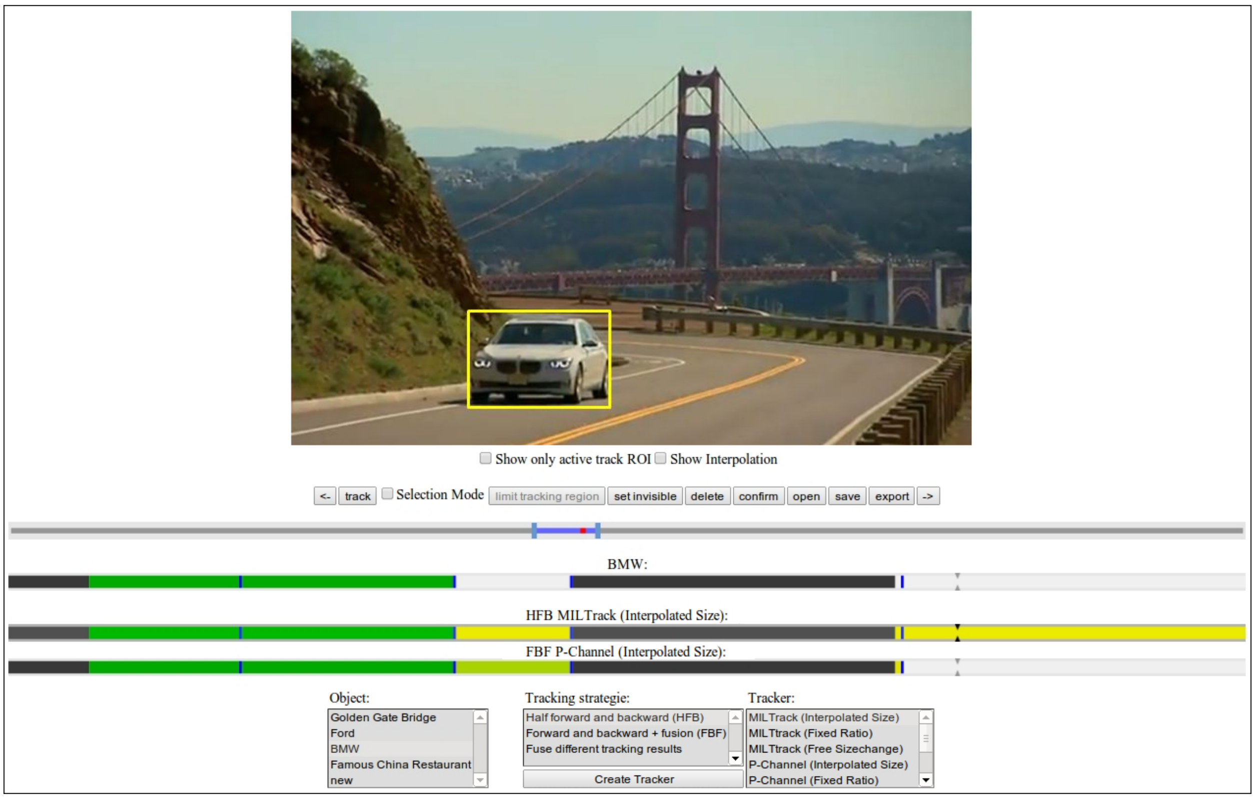
Our prototypical user interface shown in Figure 6 consists out of several parts. In the upper half, the video is rendered, and the user can select objects and see tracking results. Every user selection and tracking result is represented as a rectangular box. In the middle part we provide a video slider and different timelines. The video slider allows the user to select the video region visible in the timelines. The uppermost timeline is the main object timeline, which shows for which frames final results are available. The status of the frames is color-coded (blue if they are user set, dark green if there is a user confirmed tracking result, black if the object is not visible or white elsewhere). The subsequent timelines are tracker timelines (one timeline for each tracker). They can additionally show tracking results not yet validated by the user in yellow and bright green[1].

To check the tracking results the user can slide through the timelines. He/She can confirm correct results, which transfers them to the main timeline and delete wrong results. Furthermore, he/she can mark frames in the main timeline as "object not visible". All three actions can be executed on single frames as well as on user selected regions in the timelines. The lower left selection box allows the user to toggle between different objects in the current video and to add new objects. The middle and right selection box allow the user to add new trackers for the current object and to fuse the results of existing trackers.

4.1 Efficient validation support

A problem when confirming tracking results is that some tracking errors last only for a few frames. In that case a user has to check a huge amount of frames to make sure that there are no tracking errors. To avoid most of this effort we use a visual interface based on tracking interpolation: we interpolate the object position data from user set and user confirmed frames and compare the interpolation to the tracking result. We visually represent the similarity of the tracking result and the interpolation in a color-coded heatmap at the bottom of the user reviewed timeline. Figure 7 shows an example of such an interpolation correlation map. In the upper timeline there are some frames where interpolation and tracking result contradict (red and yellow). Thanks to interpolation it is sufficient for the user to confirm only a few conflicting frames to solve all these conflicts. Without conflicts the whole remaining sequence can be confirmed by the user. If the user

[1]Bright green is used for "reliable" frames, see section 3 for details



BMW:

HFB MILTrack (Interpolated Size):

FBF P-Channel (Interpolated Size):

Object:
Golden Gate Bridge
Ford
BMW
Famous China Restaurant
new

Tracking strategie:
Half forward and backward (HFB)
Forward and backward + fusion (FBF)
Fuse different tracking results

Tracker:
MILTrack (Interpolated Size)
MILTtrack (Fixed Ratio)
MILTtrack (Free Sizechange)
P-Channel (Interpolated Size)
P-Channel (Fixed Ratio)

Figure 6: The user interface in the browser window. The sliders are timelines. The first slider limits the range of the subsequent sliders. There, colors show the tracking status of a frame. See text for a detailed description.

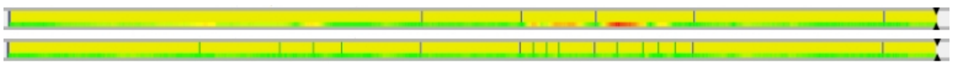


Figure 7: Interpolation support. The bottom timeline is the top timeline after conflict resolution by the user.

can not confirm a frame because the tracking result is erroneous the interpolation helps him to quickly isolate erroneous frames from correctly tracked frames. The isolated frames can then for example be tracked again with more user set frames.The interpolation is made with four Akima splines from ALGLIB [Boc14]: two for the x and y position and two for the width and height of the tracking box.

5 The complete OnEye System

In this section we describe the complete OnEye system consisting out of OnEye Creator, OnEye Designer, OnEye Videos and OnEye Player (Figure 8). The whole system consists of client-server based web applications. This is a big advantage as it for example allows a user to create an interactive video directly on the webpage were he wants to publish it.

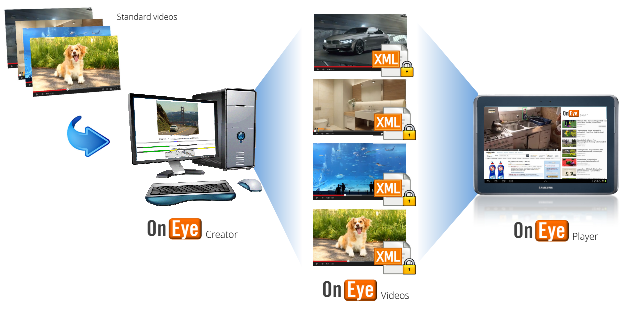


Figure 8: The OnEye system consisting out of OnEye Creator, OnEye Videos, OnEye Player and OnEye Designer (not shown).

OnEye Creator OnEye Creator is the web application that allows the user to create interactive videos. First the user has to upload the video through the web interface to the server. Then this video is available for editing through the web interface described in the last section. The user can save the processing state for a video at any time through the save button in the interface shown in Figure 6. Saved states are automatically recovered if the video is later opened again in the browser. The server application can run several sessions simultaneously. This allows one or multiple user to process multiple videos at the same time in different browser tabs and on different computers. After a video is processed it can be saved as an OnEye Video by pressing the export button (Figure 6). During export the user is asked to fill in additional metadata like weblinks for each tracked object or a description of the video.

OnEye Video An OnEye Video is a video file that contains the locations of all tracked objects in the video in each frame as supplementary information. Furthermore, it contains the metadata given for each object and the video as well as the framerate of the video. The framerate is necessary in order to synchronize the interactive content with the video playback. In most web browsers it is not possible to get the framerate of the video from the browser itself. Currently, the supplementary information is encoded as a separate XML file, but in future versions we are planing to encode it direly into the video file.

OnEye Player The OnEye Player is a HTML5-based video player that can play videos in the same manner as standard players, while providing the extra possibility to interact with objects by moving the mouse cursor over them or clicking on them. The event which is executed when moving over or clicking onto an object is defined by the OnEye Design. The standard design shows the object name when moving the cursor above an object and opens a link specified for each object in a mini-browser below the video when the object is clicked. The solution of the standard design is already very generic and allows for different kinds of content being loaded by clicking.

OnEye Designer With the OnEye Designer it is possible to create additional OnEye Designs. OnEye Designs describe the behavior of the interactive part of the OnEye Player. Currently OnEye Designs are created by the implementation of JavaScript functions. This is powerful as JavaScript allows to modify the whole webpage as desired, but it requires a programmer for the creation of designs. To make it easier to use for non-programmers the JavaScript functions may in later versions be generated by a more user friendly designer front-end. Designs must define fallbacks for OnEye Videos that do not contain all the information required by the design. Otherwise they are not accepted as finished designs by the OnEye Designer. The standard design for example shows the name of the object in the mini-browser below the object if no link is available.

6 Results

In this section we present results of the evaluation of our approach. Section 6.1 is about our main evaluation where we evaluate our tracking approach as a whole and in Section 6.2 we evaluate single tracking methods that we created for our approach.

6.1 Evaluation of our tracking approach

We tested our approach on several tracking sequences, with ground truth data available (Figure 9). In doing so, we considered frames that had an overlap over 0.8 ($\alpha = 80\%$) to the ground truth as correctly tracked. According to our visual tests this is sufficient for common interactive video applications, where no exact object boundary must be known. The parameter β is set to 4. The dynamic speed variances of the particle tracker were set to:

$$V = 0.05c\frac{O_w O_h}{2} \tag{8}$$

where O_w and O_h are the width and height of the object and c is 1 and 3, for the two variances. Similar values were set for the variances of the VTD tracker. γ is set to 15 and N to 1000.

To simulate the effect of additional user set bounding boxes we tracked each of the sequences several times with different number of preset bounding boxes taken from the ground truth data. By varying the number of preset frames, we can construct a diagram showing the number of correctly tracked frames in relation to the number of preset frames. This kind of evaluation contrasts with standard tracker evaluations in the sense that they usually reduce to one single input

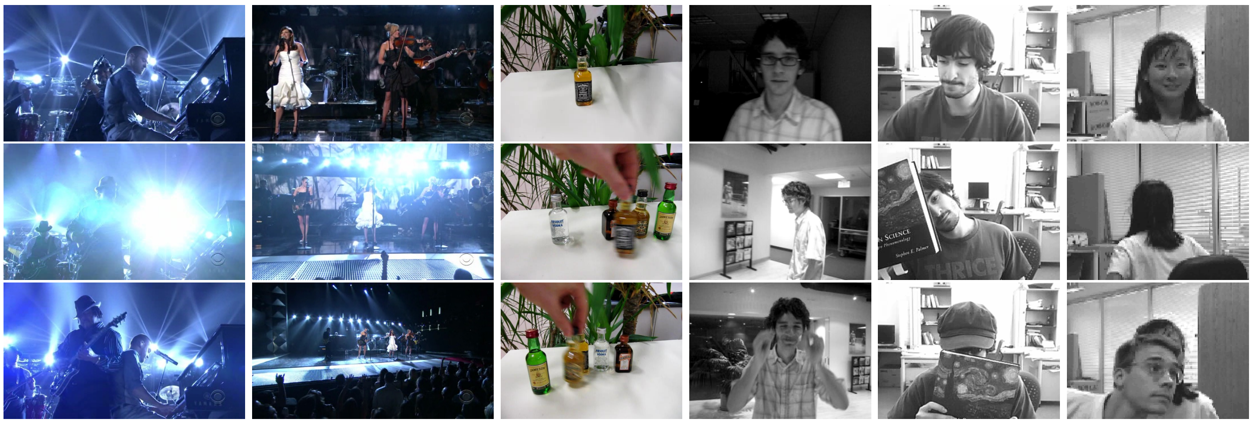


Figure 9: Frames of the tracking sequences used for evaluation. From left to right: Shaking, Singer1, Liquor, David, Faceocc2, Girl.

frame. The positions of the preset boxes in the video were set randomly, whereby we averaged over $n = 8$ runs with different random selections for each number of preset frames. In more detail: The result value R^k_c for tracker k with c preset bounding boxes is calculated as:

$$R^k_c = \frac{1}{n}\sum_{i=0}^{n-1} O^*\left(T_k(S(P^i_c)), G\right) \tag{9}$$

where P^i_c are the first c values of random permutation P^i. P^i contains all frame numbers I_G where ground truth data is available[2]. $S(P^i_c)$ is the sequence with preset ground truth data at P^i_c and $T_k(S(P^i_c))$ is the tracking result for $S(P^i_c)$ with tracker k. G is the ground truth data and O^* is calculated as:

$$O^*(T,G) = \frac{1}{|I_G|}\sum_{i\,\in I_G}\begin{cases}0 & O(T^i,G^i) < \alpha\\ 1 & \text{otherwise}\end{cases} \tag{10}$$

where T^i and G^i are a tracking and ground truth box at frame i.

The results can be seen in Figure 10.[3] The x-axis of the diagrams show the number of preset frames c, the left y-axis the percentage of correctly tracked frames R^k_c and the right y-axis the number of correctly tracked frames. The “User Set” curve shows the amount of user set frames. Thus, $y_{right} = x$ for this curve. All the sequences in Figure 10 were tracked with interpolated size change and the fusion is created out of the tracking results of all 4 trackers. For an explanation of “Fusion max.” see below. The figure shows that our automatic fusion approach works quite reliable. For many sequences it even outperforms all four trackers. Only for the girl sequence it is clearly worse than the best tracker, but anyhow still better than the second best tracker i.e. if the user does not want to validate all 4 tracking results validating the fusion is still a good choice.

As can also be seen in Figure 10 it is worth for all sequences to provide a few more than one bounding box as input, as the amount of correctly tracked frames can thereby be raised significantly e.g. 537 frames more are correctly tracked by setting 3 instead of 1 bounding box for the Liquor sequence. On the other hand it is not recommendable to preset too many bounding boxes as the advance per box is getting lower the more boxes are set. Towards 100 % the advance is even less than one box per preset box. The reason is that if a box is preset for a frame that would be tracked correctly without this preset box the advantage is zero. However, the user can simply avoid setting too many redundant boxes by iteratively tracking several times like shown in Figure 1 i.e. in the first pass he only sets a few bounding boxes, tracks with them and then knows where he has to set additional boxes for the second tracking pass, because the tracking failed there in the first pass.

As the user probably will not set bounding boxes at random frames, but will carefully select the frames he sets boxes for, we also show the fusion result for the best of the n runs as “Fusion max.” i.e. the maximum instead of the average in Equation 9. Hence, the difference of Fusion and Fusion max. gives an impres-

[2]4 datasets only have for every firth frame ground truth data available

[3]Note that we test with 80 % overlap. In tracking literature often 50 % is already considered as sufficient.

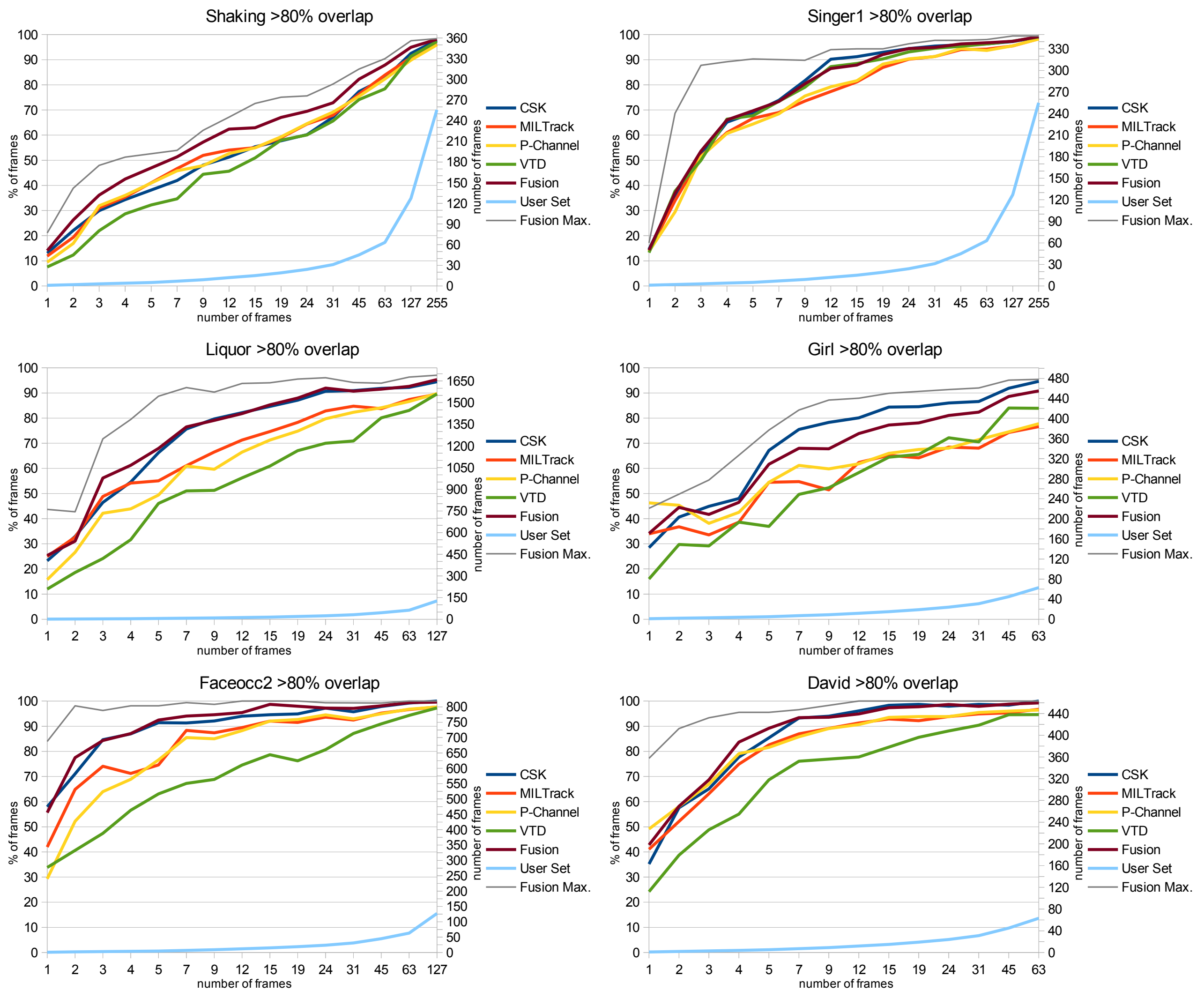


Figure 10: Tracking results for different sequences with different number of simulated user set frames. We performed 8 experiments with different randomly set frames for each tested ”number of frames“ value. Colored curves show the averages of all 8 experiments. The gray curve shows the maximum of the 8 experiments. It should be closer to a human selection than the average. See text for details.

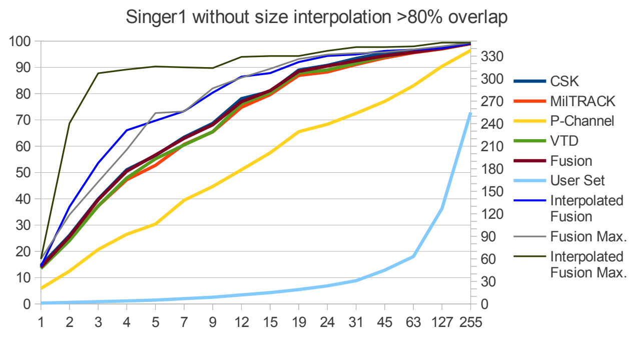


Figure 11: Singer 1 sequence without size interpolation like in Figure 10. See text for details.

sion of the impact of setting boxes at relevant instead of random frames. It is no surprise that the impact is the biggest for the Singer1 sequence as the singer undergoes strong size change and with size change it is beneficial to select positions that are good for size interpolation. Figure 11 shows the Singer1 sequence without size interpolation i.e. the tracker tracks with the size the tracking box was initialized. The result is clearly worse than with interpolation. The benefit of “Fusion Max.“ is also not as big as with interpolation. This proves that good box selection is even more important with size interpolation.

Figure 12 shows the percentage of frames that where marked as ”reliable“ by a tracker [R] and the

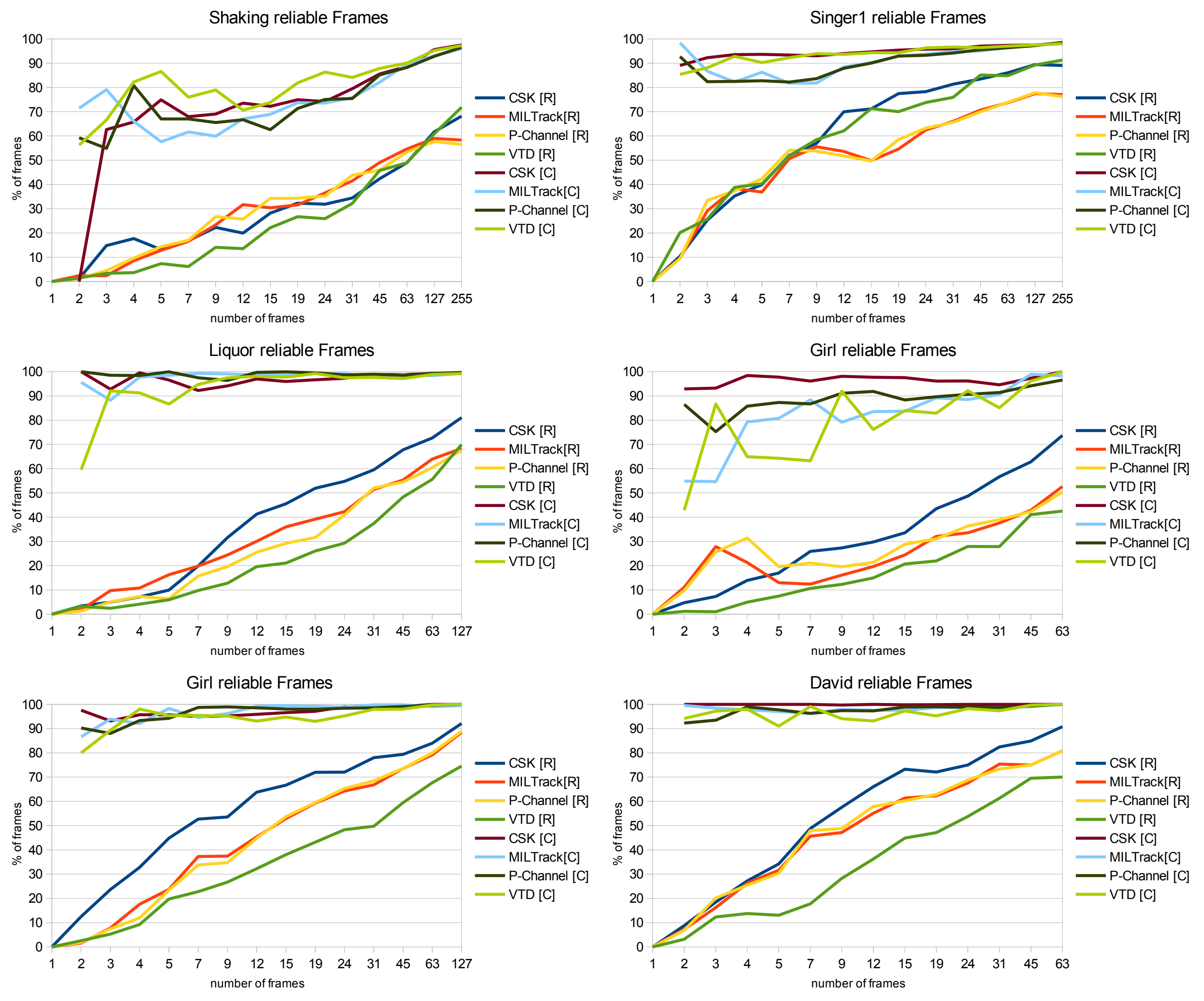


Figure 12: The percentage of frames that are labeled as ”reliable“ [R] and the percentage of ”reliable“ labeled frames that are correct [C]. See text for details.

percentage of ”reliable“ marked frames that are correct [C] i.e. have an overlap of at least α to the ground truth. To calculate the percentage of ”reliable“ frames we exclude user set frames i.e. 100 % are all non user set frames. As there are no ”reliable“ frames for $c = 1$, [C] can not be determined there. The figure shows that our reliability measure is reliable for most sequences. Only for two trackers of the Girl and the trackers of the Shaking sequence the reliability is not very high. Anyhow the shaking sequence has the best fusion result. Moreover, it is interesting to see that in sequences where the reliability measure worked well it even worked well if only very few frames where marked as ”reliable“.

Note that the curves of the diagrams of Figure 10 can not directly be compared to the curves of Figure 12. To compare them the diagrams in Figure 10 have to be stretched so that the ”User Set“ curve is on the zero line, as this excludes user set boxes. However, because the ”User set“ curve is close to zero for most data points direct visual comparison is anyhow possible.

6.2 Evaluation of our tracking methods

Here we provide a short evaluation of our particle based tracking methods using on our P-Channel and MILTrack implementations. Results can be seen in Figure 13. We did not only evaluate MILTrack with HAAR+HOG features, but we also tested HAAR and HOG independently with our particle based motion model. Furthermore, we tested color HOG features (CHOG) on the Lemming and Liquor sequences which

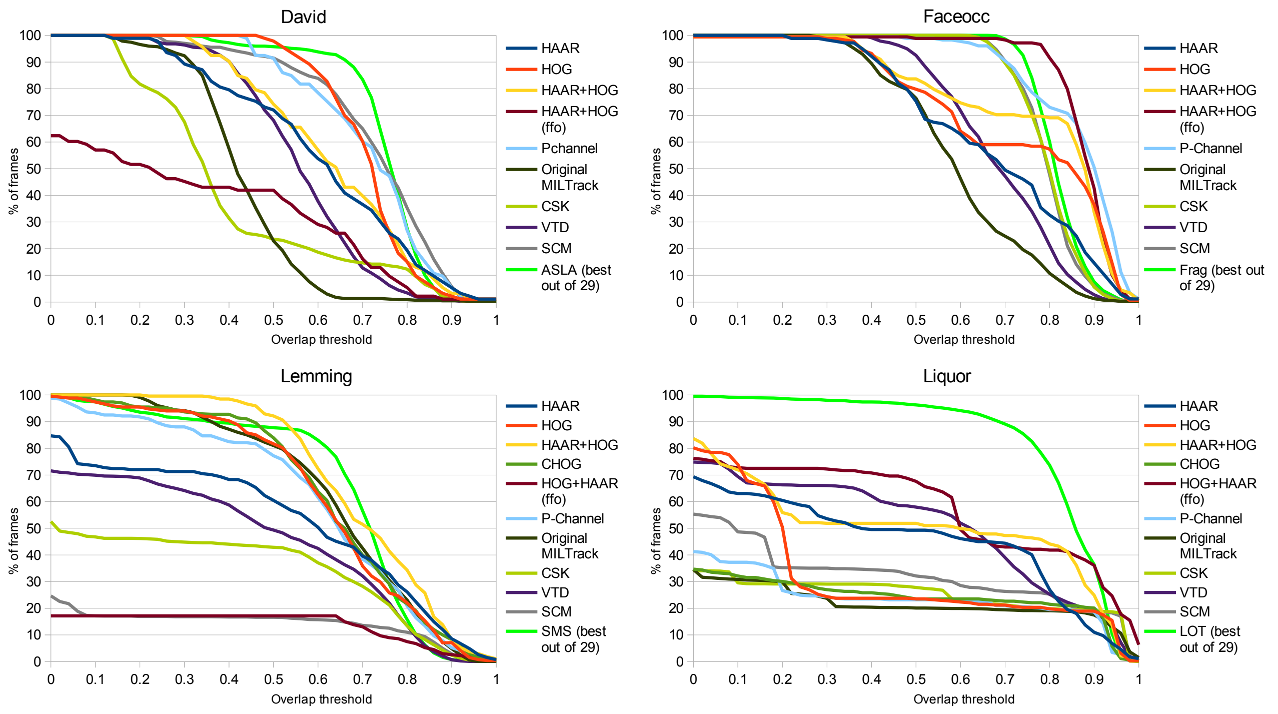


Figure 13: Evaluation of our tracking methods. Our particle based methods can compete well even with the best state-of-the-art method for a sequence. SCM is the method that performs in average best on 51 sequences (See [WLY13]). The bight green method is the best method out of 29 on the given sequence.

are in color. As P-Channel representations are very robust against appearance changes our P-Channel implementation does not perform appearance update. This avoids drifting caused by of model update. In contrast, MILTrack is updated every frame as the method is very resistant against drift by the model update. However, we additionally evaluated MILTrack without model update (ffo)[4]. For comparability we also show CSK, VTD, SCM and the method that works best for a sequence out of 29 methods – including state of the art.[5] Results for SCM and the best method for each sequence are taken from the evaluation of Wu et al. [WLY13]. SCM is the method that performed in average best on all the sequences they tested. Results are created by simple forward tracking from the first frame. This allows us to show not only one overlap threshold like in the other figures but all thresholds at the same time.

As Figure 13 shows HOG features perform on average better than HAAR features, but HAAR+HOG is on average even better. On 3 of the 4 sequences HAAR+HOG outperforms HAAR as well as HOG. HAAR+HOG (ffo) sometimes performs better than HAAR+HOG with model update, but if it performs worse the performance is very poor. P-Channel can compete well with HAAR+HOG but shows different strengths. CHOG does not give a big benefit over HOG, but can easier be irritated in colorful sequences like Liquor.

Our particle based MILTrack implementation with HAAR+HOG features outperforms the original MILTrack in all 4 sequences – in 3 of them even by far. Furthermore, on 3 sequences our best method for the sequence can compete well with or even outperform the best performing method out of the set of 29 methods described above. This shows that our particle based model with two different spread variances is very robust and that our methods can compete very well with the state of the art, thanks to the model.

[4]ffo = first frame only

[5]According to average overlap, which is the area under curve in the plots in Figure 13.

7 Conclusion

In this paper we presented a complete interactive video system called OnEye. The main focus was set to our powerful semi-automatic tracking framework for interactive video production. In this context, we developed new tracking methods like our powerful particle tracker. Furthermore, we showed that we can significantly improve the tracking result by using only a few more user provided bounding boxes. Thanks to the validation support it is possible to validate tracking results very quickly, so that the user can go on setting additional bounding boxes at positions where tracking failed. We think this iterative way of first setting boxes, then tracking and finally confirming the result is a very efficient way of creating interactive videos despite unreliable tracking algorithms. To further improve the tracking result with multiple bounding boxes we introduced the ideas of forward/backward tracking, size interpolation and a robust reliability measure. Our fusion approach, which utilizes the reliability measure, created good results for all of our tests. In many situations it even outperformed the best tracker. We showed that this approach is a good and much faster alternative compared to manual fusion of different tracking results. Altogether our framework is a very fast way for semi-automatic tracking and thus for creating interactive videos.

Acknowledgement

This work was partially funded by the Eurostars-Project VIDP under contract number E!5558 (FKZ 01AE1103B, BMBF 01QE1103B).

References

[Ber12] Pascal Bertolino, *Sensarea: An authoring tool to create accurate clickable videos*, 10th International Workshop on Content-Based Multimedia Indexing (CBMI 2012), IEEE, 2012, DOI 10.1109/CBMI.2012.6269804, pp. 1–4, ISBN 978-1-4673-2368-0.

[Boc14] Sergey Anatolyevich Bochkanov, *ALGLIB Project*, www.alglib.net, 2014, Last visited October 23rd, 2014.

[BYB11] Boris Babenko, Ming-Hsuan Yan, and Serge Belongie, *Robust object tracking with online multiple instance learning*, IEEE Transactions on Pattern Analysis and Machine Intelligence **33** (2011), no. 8, 1619–1632, ISSN 0162-8828, DOI 10.1109/TPAMI.2010.226.

[CAG12] Manisha Chate, S. Amudha, and Vinaya Gohokar, *Object Detection and tracking in Video Sequences*, ACEEE International Journal on signal & Image processing **3** (2012), no. 1, 36–41.

[HCMB12] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, *Exploiting the Circulant Structure of Tracking-by-detection with Kernels*, Computer Vision – ECCV 2012, Lecture Notes in Computer Science Vol. 7575, 2012, DOI 10.1007/978-3-642-33765-9_50, pp. 702–715, ISBN 978-3-642-33764-2.

[KC11] Dominik A. Klein and Armin B. Cremers, *Boosting scalable gradient features for adaptive real-time tracking*, 2011 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, DOI 10.1109/ICRA.2011.5980369, pp. 4411–4416, ISBN 978-1-61284-386-5.

[KL10] Junseok Kwon and Kyoung Mu Lee, *Visual tracking decomposition*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010), IEEE, 2010, DOI 10.1109/CVPR.2010.5539821, pp. 1269–1276, ISBN 978-1-4244-6984-0.

[MIB04] Iain Matthews, Takahiro Ishikawa, and Simon Baker, *The template update problem*, IEEE transactions on pattern analysis and machine intelligence **26** (2004), no. 6, 810–815, ISSN 0162-8828, DOI 10.1109/TPAMi.2004.16.

[PSF09] Alain Pagani, Didier Stricker, and Michael Felsberg, *Integral P-channels for fast and robust region matching*, 16th IEEE International Conference on Image Processing (ICIP 2009), IEEE, 2009, DOI 10.1109/ICIP.2009.5414467, pp. 213–216, ISBN 978-1-4244-5653-6.

[VG12] Sudheendra Vijayanarasimhan and Kristen Grauman, *Active frame selection for label propagation in videos*, Computer Vision – ECCV 2012, Lecture Notes in Computer Science Volume 7576, 2012, DOI 10.1007/978-3-642-33715-4_36, pp. 496–509.

[VPR13] Carl Vondrick, Donald Patterson, and Deva Ramanan, *Efficiently scaling up crowdsourced video annotation*, International Journal of Computer Vision **101** (2013), no. 1, 184–204, ISSN 0920-5691, DOI 10.1007/s11263-012-0564-1.

[VR11] Carl Vondrick and Deva Ramanan, *Video Annotation and Tracking with Active Learning*, Advances in Neural Information Processing Systems 24 (J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, eds.), Curran Associates, Inc., 2011, pp. 28–36.

[WCXY11] Qing Wang, Feng Chen, Wenli Xu, and Ming-Hsuan Yang, *An experimental comparison of online object-tracking algorithms*, SPIE: Image and Signal Processing **8138** (2011), 81381A–81, DOI 10.1117/12.895965.

[WLY13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, *Online object tracking: A benchmark*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013), IEEE, 2013, DOI 10.1109/CVPR.2013.312, pp. 2411–2418.

[YRLT09] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba, *Labelme video: Building a video database with human annotations*, IEEE 12th International Conference on Computer Vision (ICCV 2009), IEEE, 2009, DOI 10.1109/ICCV.2009.5459289, pp. 1451–1458, ISBN 978-1-4244-4420-5.

[YSZ+11] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song, *Recent advances and trends in visual tracking: A review*, Neurocomputing **74** (2011), no. 18, 3823–3831, ISSN 0925-2312, DOI 10.1016/j.neucom.2011.07.024.

[Ziv04] Zoran Zivkovic, *Improved adaptive Gaussian mixture model for background subtraction*, Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004), vol. 2, IEEE, 2004, DOI 10.1109/ICPR.2004.1333992, pp. 28–31, ISBN 0-7695-2128-2.

[ZLY12] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang, *Robust object tracking via sparsity-based collaborative model*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012), IEEE, 2012, DOI 10.1109/CVPR.2012.6247882, pp. 1838–1845, ISBN 978-1-4673-1226-4.

Citation

Christian Bailer, Alan Pagani, and Didier Stricker, *A user supported object tracking framework for interactive video production*, Journal of Virtual Reality and Broadcasting, 11(2014), no. 9, December 2014, urn:nbn:de:0009-6-40305, ISSN 1860-2037.